Vehicle Insurance Claim Fraud Detection

Mohammad Rouintan

Computer Science, Shahid Beheshti University

Abstract

Vehicle insurance fraud involves conspiring to make false or exaggerated claims involving property damage or personal injuries following an accident. Some common examples include staged accidents where fraudsters deliberately "arrange" for accidents to occur; the use of phantom passengers where people who were not even at the scene of the accident claim to have suffered grievous injury, and make false personal injury claims where personal injuries are grossly exaggerated. We implemented different classification algorithms on the dataset. Their performances are recorded and compared. Hyperparameter Tuning with GridSearchCv or RandomizedSearchCv are used to improve performance.

Introduction

We are going to work with the Vehicle Insurance Claim Fraud Detection dataset. We will implement multiple classification models using the Scikit-Learn package to predict if a claim application is fraudulent or not, based on about 32 features. This dataset is imbalanced so In this way we use kind of techniques to overcome this issue. for example:

- UnderSampling
- OverSampling
- WeightBased Approaches

In the other hand we use GridSearchCv for tune our model's hyperparameters . We can also measure the result of the model using different metrics and adjust the model so that it is the best model according to our desired metrics. For example, we can use the following metrics:

- Accuracy Score
- F1 Score
- Recall Score
- Precision Score
- ROC AUC

According to the dataset and the purpose of the project, we can choose the appropriate metric. For example, in this dataset, our goal is to detect vehicle insurance fraud, so our overall goal is to detect all frauds, which means it is not a problem even if a small amount of data is wrongly identified as fraud. But we must be careful that fraud does not get out of hand. In fact, we want to have a high Recall. In this project, we measure our models based on the all metrics and find best models.

Methodologies

Exploratory Data Analysis

first, for data analysis we should read description of dataset and know features and target of dataset in detail. This dataset contains vehicle dataset - attribute, model, accident details, etc along with policy details - policy type, tenure etc. The target is to detect if a claim application is fraudulent or not ***FraudFound_P***.

According to the columns of the dataset, we understand that this dataset has 8 numerical features and the rest of the features are categorical.

Target: First, we evaluate the target (FraudFound_P) and find that the number of zero targets (No Fraud Claim) is 14497 and the number of one targets (Fraud Claim) is 923.

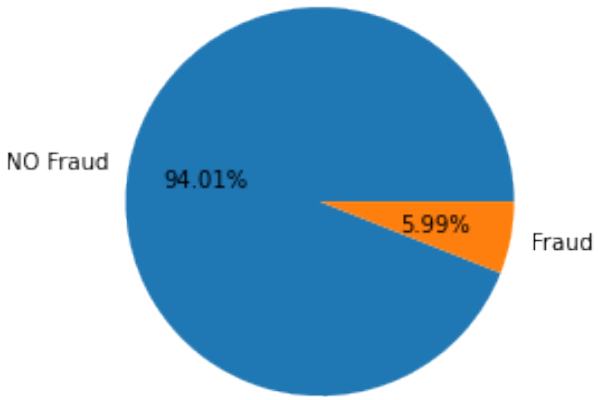

Figure 1: *No Fraud : 94.01% , Fraud : 5.99%*

From the pie plot, we understand that this dataset is imbalanced, that means we don't have enough samples from each class and we will have problems to train the model, so we have to balance this dataset before training the model. We do this in the data preprocessing section.

Time features: In this section we analyze Month, WeekOfMonth and DayOfWeek of samples. The

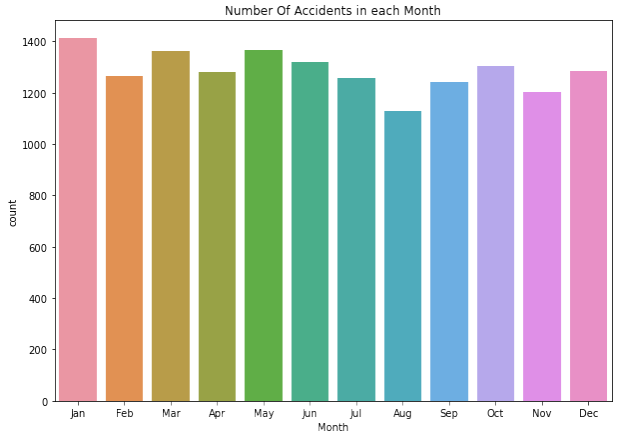

Figure 2: *January: 1411 number of claims (Highest), August: 1127 number of claims (Lowest)*

amount of claims in different months does not differ so much that we can find out a special point from it.

Also, on average and regardless of different months, the most claims belong to the following weeks respectively:

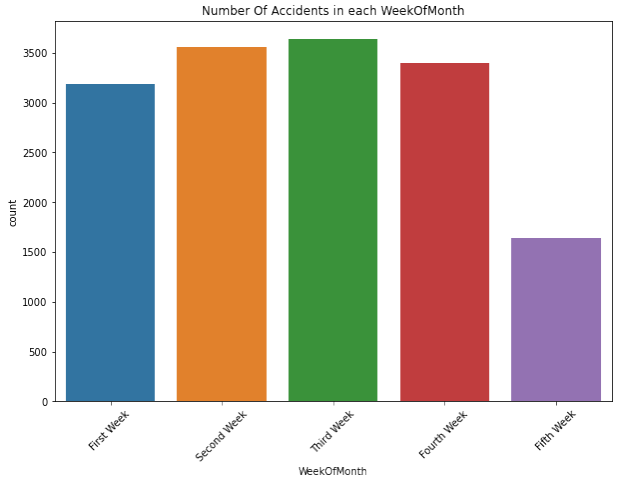

Figure 3: *1. Third Week (Highest), 2. Second Week, 3. Fourth Week 4. First Week, 5. Fifth Week (Lowest)*

It is noteworthy that in all months, most of the claims were in the middle weeks of the months. As we can see, in all months, the most claims were in the second, third or fourth week (Figure 4). As you can see, most of the claims are on Monday, Tuesday, Thursday and Friday, that is, on the first day of the week and the last working day of the week (Figure 5).

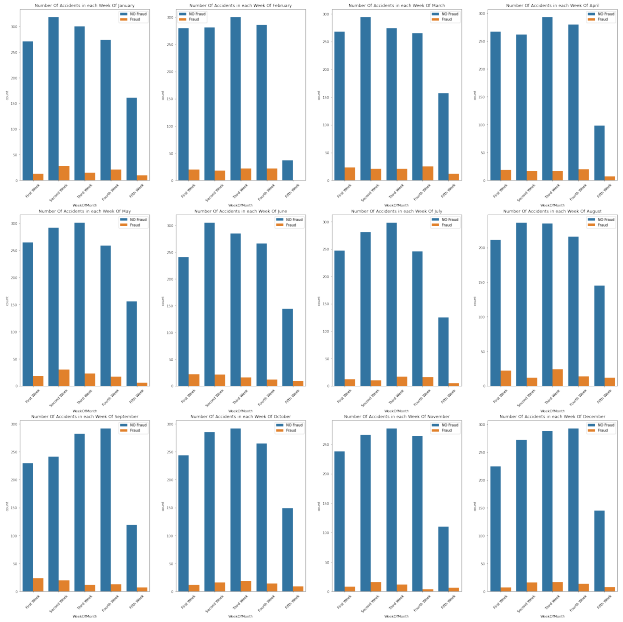

Figure 4: *Analyse weeks of each month*

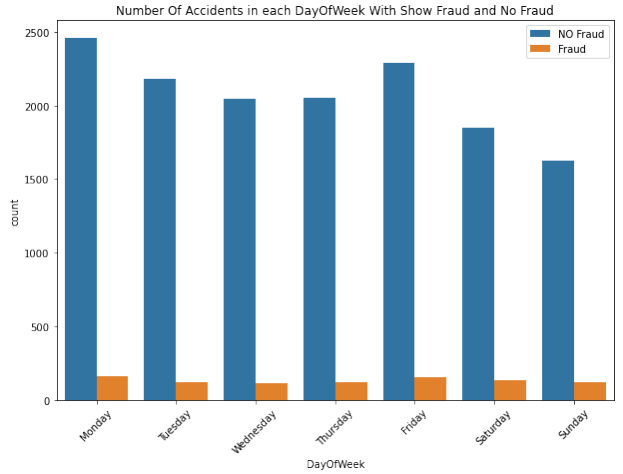

Figure 5: *In order, the most claims belong to days: 1. Monday (Highest), 2. Friday, 3. Tuesday, 4. Thursday, 5. Wednesday, 6. Saturday, 7. Sunday (Lowest)*

Human features: In this section we analyze Sex, MaritalStatus, Age, AgeOfPolicyHolder and Fault of samples (Figures 6, 7, 8, 9, 10, 11).

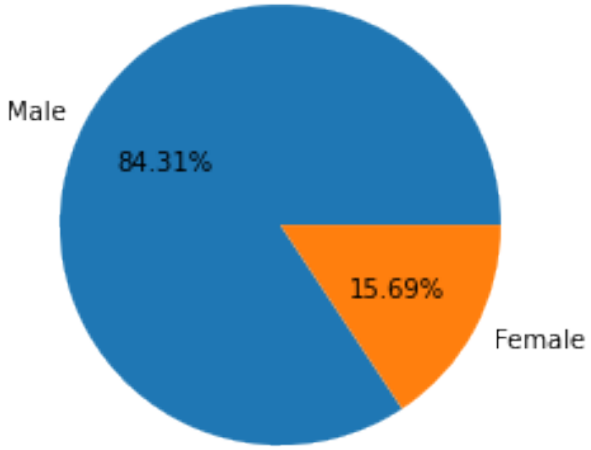

Figure 6: *Most driver are men (approximately 84.31%) and the remaining 15.69% are women.*

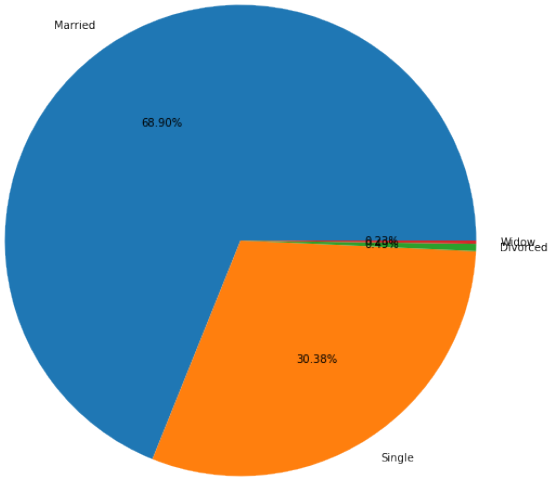

Figure 7: *Most of the claims are from people who are married (about 68.90 percent) and about 30.38 percent of the claims are from single people. The remaining percentage is from people who are either divorced or widow.*

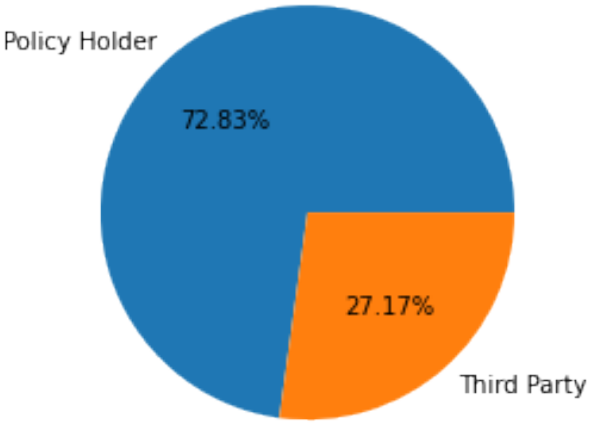

Figure 11: *About 72.38% of the injured are insurance policy holders. The remaining 27.17% are third party people. But according to the histogram,*

Vehicle features: In this section we analyze VehicleCategory, VehiclePrice, AgeOfVehicle, NumberOfCars and PolicyType of samples. By evaluating the Vehicle Category, we see that most of the claims were for sedan cars, followed by sports cars and finally utility cars. In the following order (Figure 12):

1. Sedan (62.72 %)
2. Sport (34.75 %)
3. Utility (2.54 %)

In the next analysis, which is related to the price range of Vehicle, most of the claims are related to the following price ranges (Figure 13):

1. 20000 to 29000 (Highest)
2. 30000 to 39000
3. more than 69000
4. less than 20000
5. 40000 to 59000
6. 60000 to 69000 (Lowest)

In the next analysis, which is related to the age of vehicle, top 4 of the claims are related to the following age of vehicle (Figure 14):

1. 7 years (Highest)
2. more than 7 years
3. 6 years
4. 5 years

In most of the claims, the number of cars was one (Figure 15). In the last analysis, which is related to the Policy Type, we see that most of the claims are related to Sedan - Collision, Sedan - All Perils and Sedan - Liability (Figure 16).

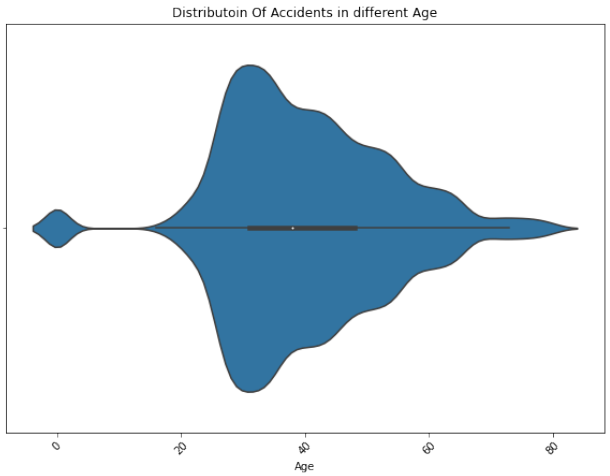

Figure 8: *According to the violin plot that displays the data distribution, we see that a large percentage of claimants are between 35 and 60 years old.*

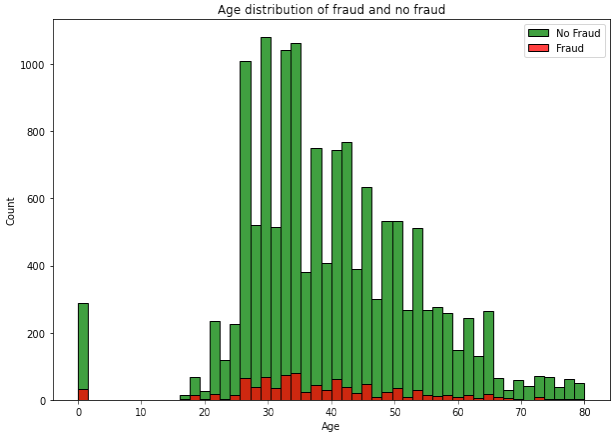

Figure 9: *Distribution of Age. The interesting thing to note is that the age of some people has been recorded as 0, which means that these samples are missing values, and these samples should either be deleted or corrected with average or median methods in the data preprocessing section.*

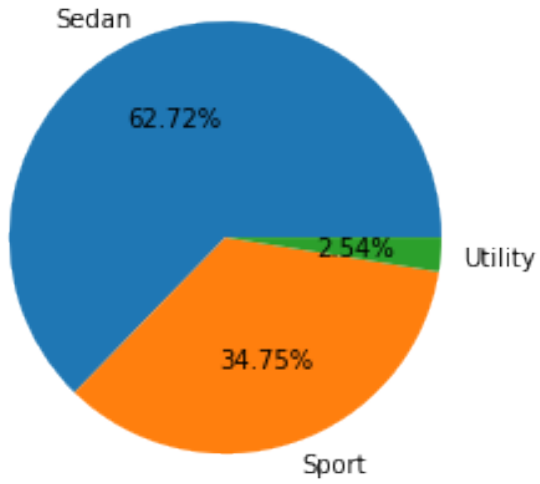

Figure 12: *VehicleCategory Analysis*

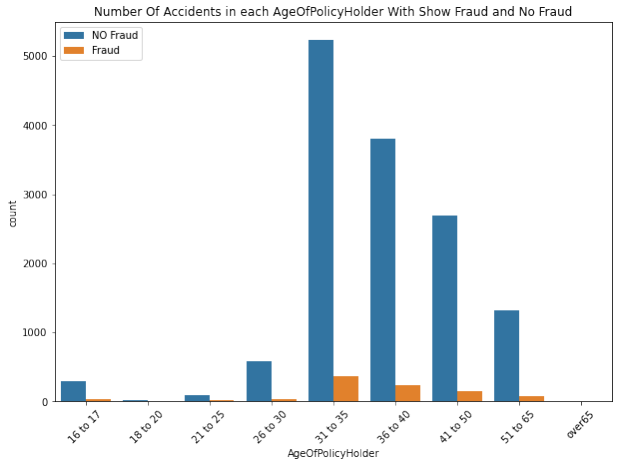

Figure 10: *The top 4 age range of policy holders who claim is as follows: 1. 31 to 35 (Highest), 2. 36 to 40, 3. 41 to 50, 4. 51 to 65*

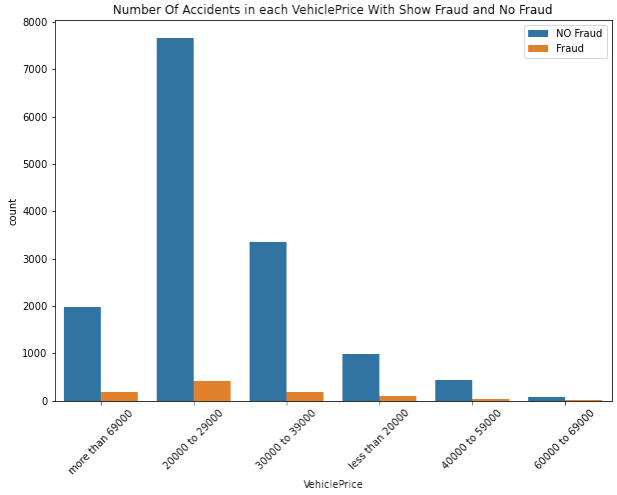

Figure 13: *VehiclePerice Analysis*

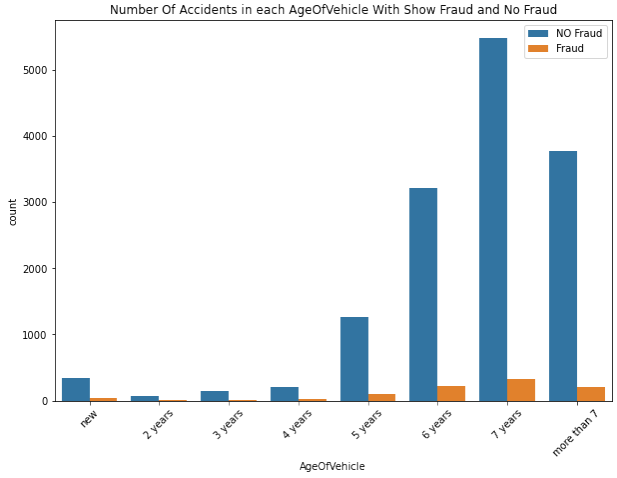

Figure 14: *AgeOfVehicle Analysis*

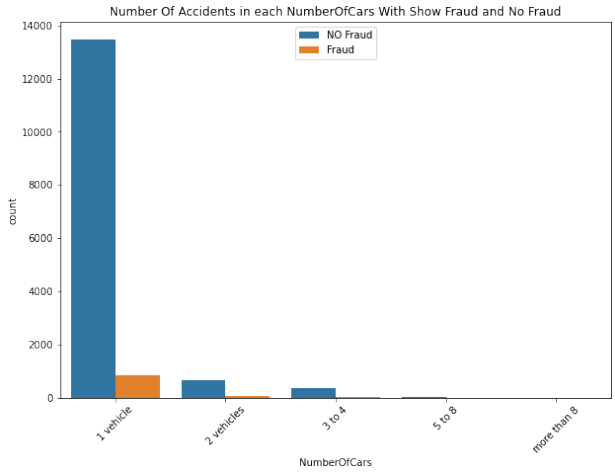

Figure 15: *NumberOfCars in Accidents Analysis*

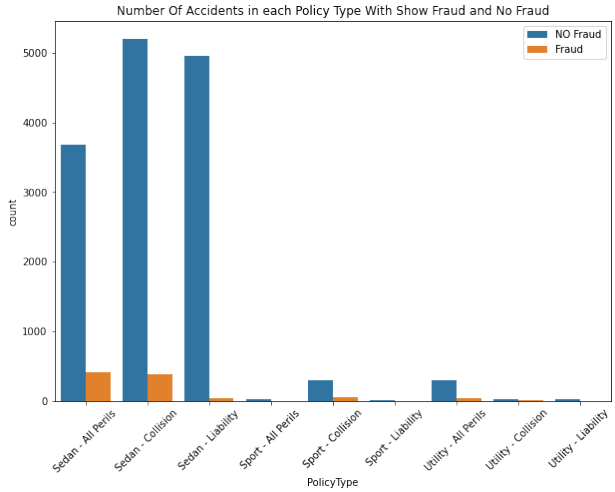

Figure 16: *PolicyType Analysis*

More information and analysis is available in the notebook.

Preprocessing Data

In data preprocessing, we first check whether there is a missing value in the dataset or not. As we said in the EDA section, in the number of samples, the age of people is entered as zero, which is not logically correct, that's why we fill the ages that are entered as zero with the mean of age of the policy holder of these samples. There is also a sample that has DayOfWeekClaimed and MonthClaimed zero entered, so we will delete this sample.

One of the data columns is called policy number, which is equal to the number of rows, that's why it cannot have a special effect on the train our model and may even cause the model to fail, so we delete this column. Then, we encode all the categorical column.

In the next step, I will divide the dataset into two parts, train and test. In this project, 20% of the data is used for testing and the rest for train. Because this data is imbalanced, we must use techniques to balance the data. These techniques include oversampling and undersampling.

OverSampling: Random oversampling involves randomly duplicating examples from the minority class and adding them to the training dataset.

UnderSampling: Random undersampling involves randomly selecting examples from the majority class to delete from the training dataset.

Training Models

In this section, we check the best result of each model and the rest of the results are checked in the notebook.

Logistic Regression: Before the training this model we use Standard Scaler for Scaling data. The best logistic regression model is when we use the oversampling technique.

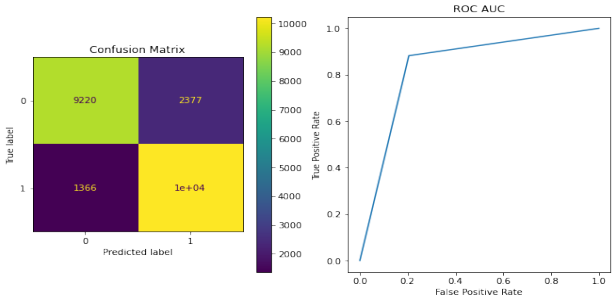

Figure 17: *F1Score Train = 0.8467738635898695 , AUC Train = 0.8385358282314391*

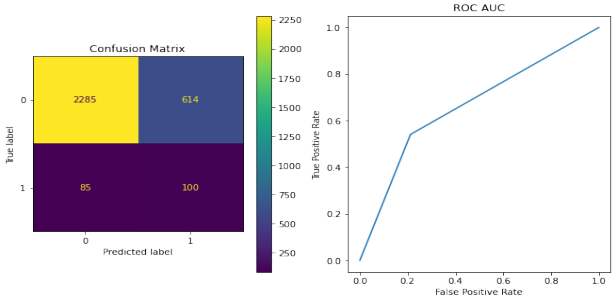

Figure 18: *F1Score Test = 0.2224694104560623 , AUC Test = 0.6643716845510568*

Linear SVC: Before the training this model we use Standard Scaler for Scaling data. The best Linear SVC model is when we use the oversampling technique.

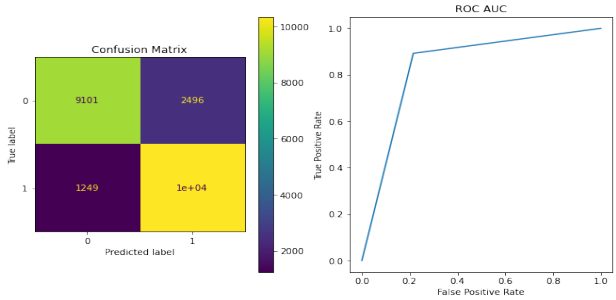

Figure 19: *F1Score Train = 0.8453625284032225 , AUC Train = 0.8386220574286454*

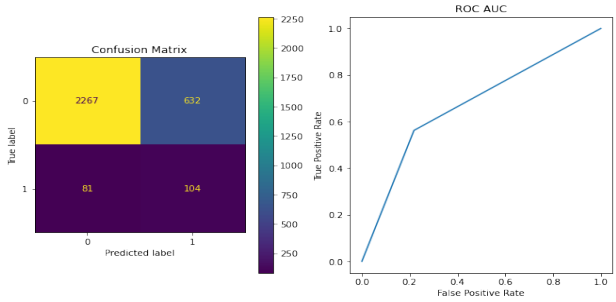

Figure 20: *F1Score Test = 0.22584147665580892 , AUC Test = 0.6720779765622815*

Polynomial SVC: Before the training this model we use Standard Scaler for Scaling data. The best Polynomial SVC model is when we use the oversampling technique.

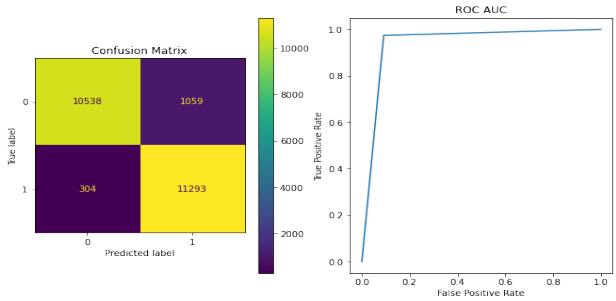

Figure 21: *F1Score Train = 0.9430873940456804 , AUC Train = 0.9412348021039925*

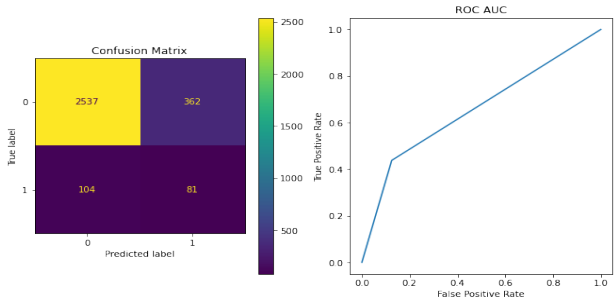

Figure 22: *F1Score Test = 0.25796178343949044 , AUC Test = 0.6564835963939103*

RBF SVC: Before the training this model we use Standard Scaler for Scaling data. The best RBF SVC model is when we use the undersampling technique.

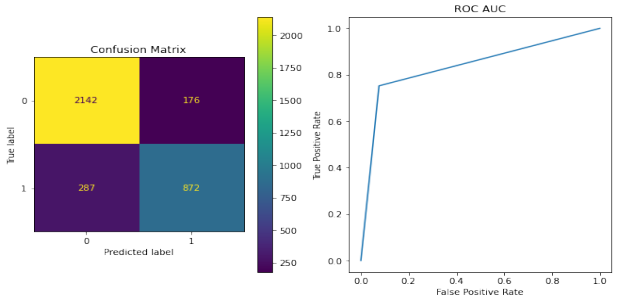

Figure 23: *F1Score Train = 0.7902129587675577 , AUC Train = 0.8382226056945643*

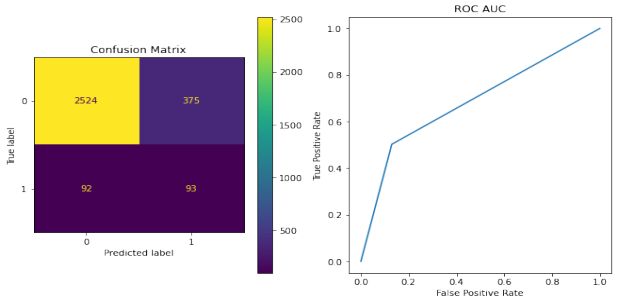

Figure 24: *F1Score Test = 0.28483920367534454 , AUC Test = 0.686673876359975*

Decision Tree: The best Decision Tree model is when we use the undersampling technique. with ***GridSearch Cv*** we find best model with hyperparameter ('criterion': 'gini', 'max_depth': 2)

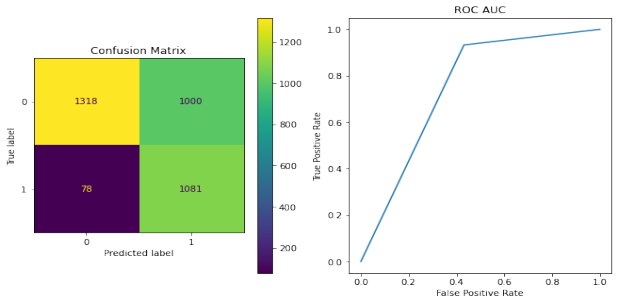

Figure 25: *F1Score Train = 0.667283950617284 , AUC Train = ROC AUC: 0.7506471095772217*

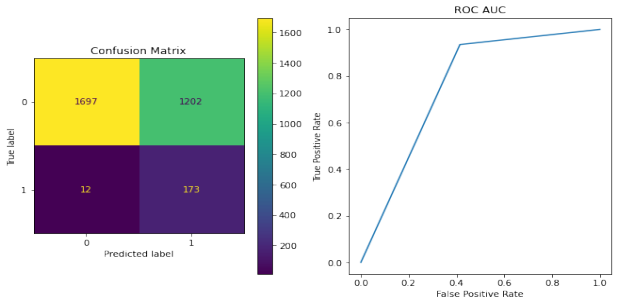

Figure 26: *F1Score Test = 0.2217948717948718 , AUC Test = 0.7602547010618761*

Random Forest: The best Random Forest model is when we use the undersampling technique. with ***GridSearch Cv*** we find best model with hyperpa-

rameter ('class_weight': 'balanced_subsample', 'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'n_estimators': 35)

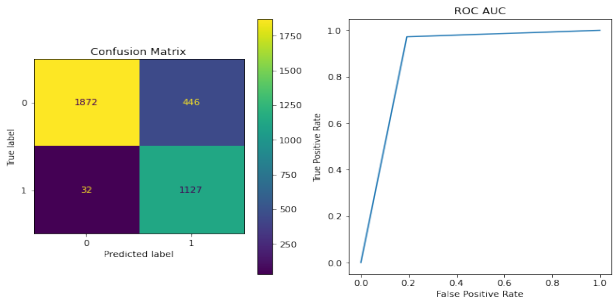

Figure 27: *F1Score Train = 0.8250366032210835 , AUC Train = 0.8899913718723037*

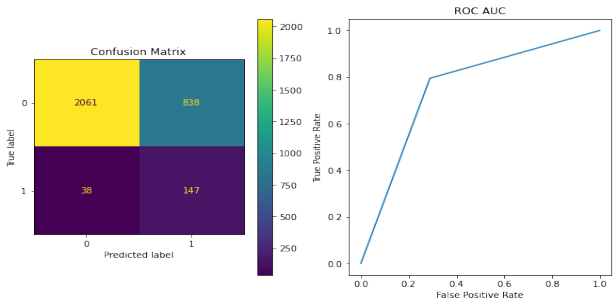

Figure 28: *F1Score Test = 0.2512820512820513 , AUC Test = 0.7527646998499016*

Extra Trees: The best Random Forest model is when we use the undersampling technique. with ***GridSearch Cv*** we find best model with hyperparameter

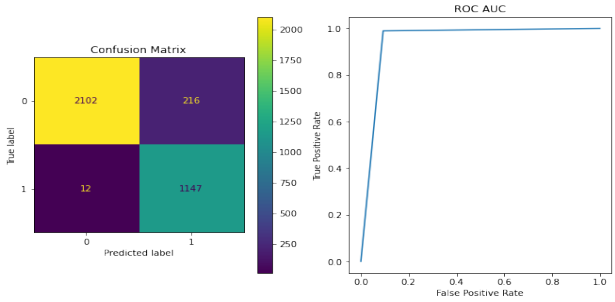

Figure 29: *F1Score Train = 0.9095955590800953 , AUC Train = 0.9482312338222605*

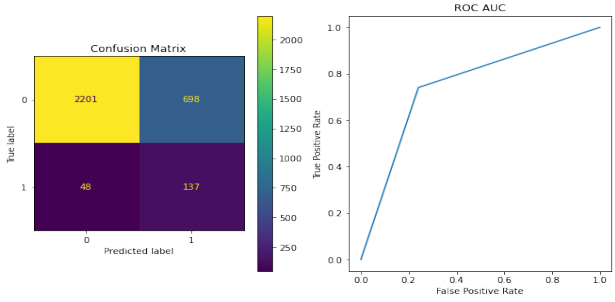

Figure 30: *F1Score Test = 0.26862745098039215 , AUC Test = 0.7498839301529884*

KNN: Before the training this model we use Standard Scaler for Scaling data. The best KNN model is when we use the oversampling technique.

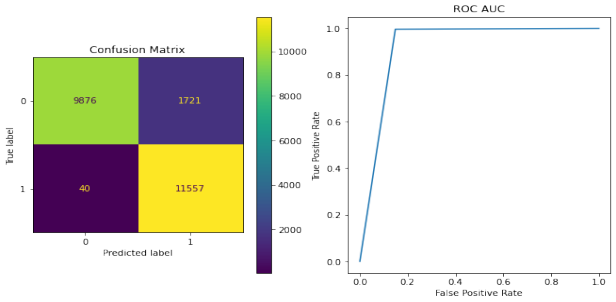

Figure 31: *F1Score Train = 0.9292060301507538 , AUC Train = 0.9240751918599636*

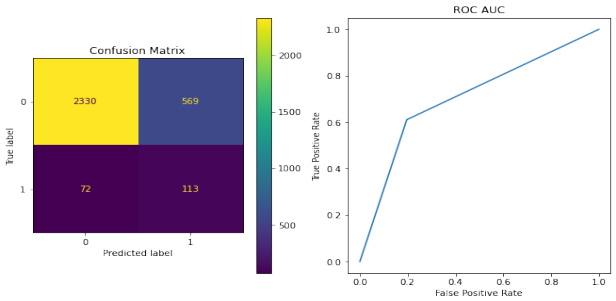

Figure 32: *F1Score Test = 0.2606689734717416 , AUC Test = 0.7072681166851571*

NaiveBayes: Before the training this model we use Standard Scaler for Scaling data. The best Gaussian NB model is when we use the undersampling technique.

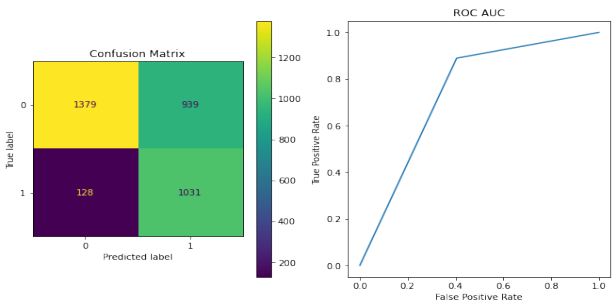

Figure 33: *F1Score Train = 0.6589964844998402 , AUC Train = 0.7422346850733391*

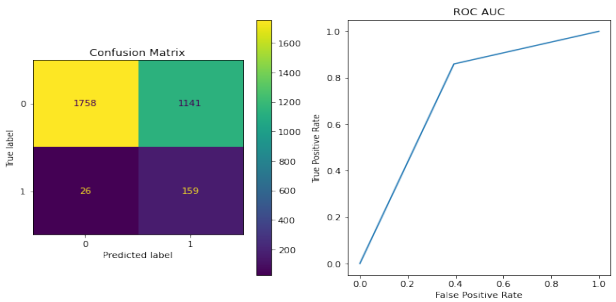

Figure 34: *F1Score Test = 0.21414141414141416 , AUC Test = 0.732937732489302*

AdaBoost: The best AdaBoost model is when we use the undersampling technique.

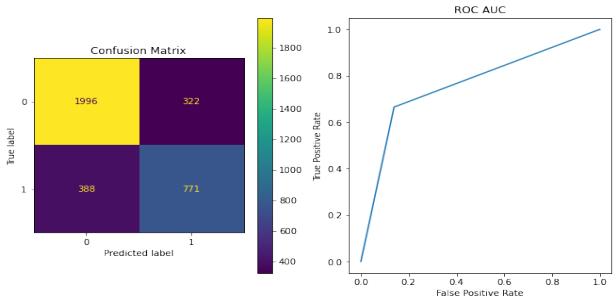

Figure 35: *F1Score Train = 0.6847246891651865 , AUC Train = 0.763157894736842*

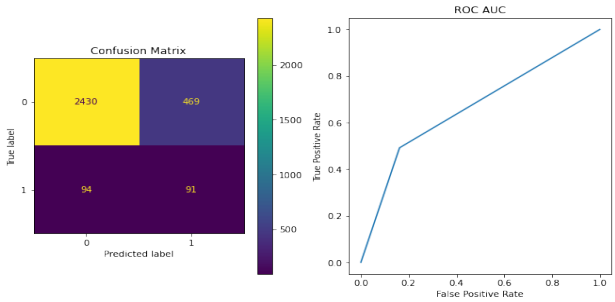

Figure 36: *F1Score Test = 0.2442953020134228 , AUC Test = 0.6650559838900646*

Gradient Boosting: The best Gradient Boosting model is when we use the undersampling technique.

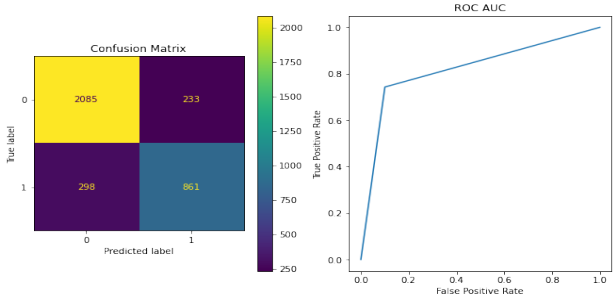

Figure 37: *F1Score Train = 0.7643142476697736 , AUC Train = 0.8211820534943918*

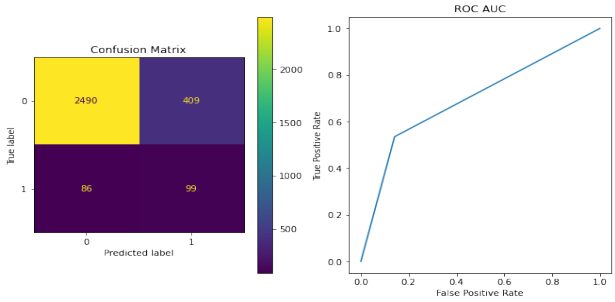

Figure 38: *F1Score Test = 0.2857142857142857 , AUC Test = 0.6970260015103064*

XGBoost: The best XGBoost model is when we use the undersampling technique.

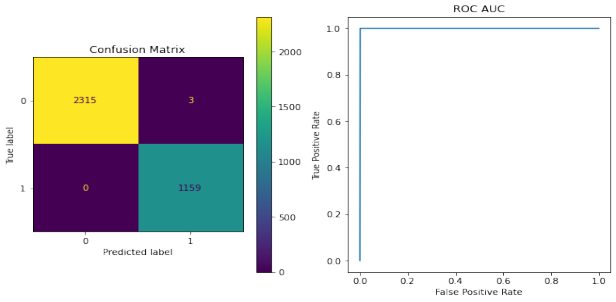

Figure 39: *F1Score Train = 0.9987074536837569 , AUC Train = 0.9993528904227782*

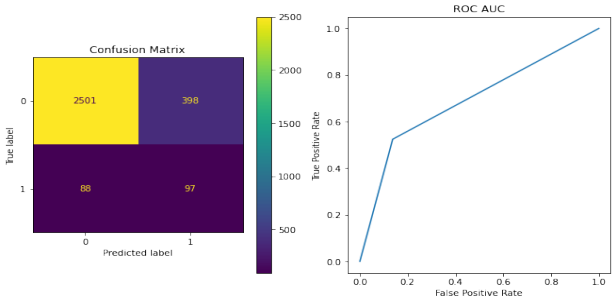

Figure 40: *F1Score Test = 0.2852941176470588 , AUC Test = 0.6935178020379814*

Results

The main purpose of the project is to examine a dataset on the subject of vehicle insurance fraud, which we should be able to detect between fraud and non-fraud using classification.

As I can see in the Training Model section, we have used 11 different classification models. We also said that we can use different metrics to measure the model and express the best results based on the desired metric. Based on the F1 score, the best models are Gradient Boosting and XGBoost, which have an F1 score of 0.29 on the test data. After that, Extra Tree, KNN and Random Forest models are also good. If we measure based on ROC AUC, the best model is Decision Tree that has 0.76 AUC in the test data. After that, the Random Forest and Extra Trees models are also good. Also if we measure based on Recall, the best model is Decision Tree that has 0.94 Recall in the test data. After that, the Naive Bayes model is also good.

In general, in order to get better results, Grid Search Cv should be done on all these models so that the best results can be obtained. Also, in the data preprocessing section, better results can be achieved by changing the procedure, for example, because this data has many categorical features, encoding these features in different ways can be effective.

Table 1: *Compare All Models*

Model	Accuracy	F1	Recall	Precision	ROC AUC
Logistic Regression	0.77	0.22	0.54	0.14	0.66
Linear SVC	0.76	0.23	0.56	0.14	0.67
Polynomial SVC	0.85	0.26	0.44	0.18	0.66
RBF SVC	0.85	0.28	0.50	0.19	0.69
Decision Tree	0.61	0.22	0.94	0.13	0.76
Random Forest	0.72	0.25	0.79	0.15	0.75
Extra Trees	0.76	0.27	0.74	0.16	0.75
KNN	0.79	0.26	0.61	0.17	0.70
Naive Bayes	0.62	0.21	0.86	0.12	0.73
Ada Boost	0.82	0.24	0.49	0.16	0.67
Gradient Boosting	0.84	0.29	0.54	0.19	0.70
XG Boost	0.84	0.29	0.52	0.20	0.69

References

[1] Preprocessing

[2] Visualization

[3] Classification

[4] Imbalanced Data

[5] Dataset codes in kaggle